A TMS320C6701/FPGA Based Frequency Selective RF Channel Simulator Using IF Sampling

Jeff R. Papenfuss
Xircom Wireless Technology Group
jpapenfuss@omnipoint.com

Mark A.Wickert
University of Colorado at Colorado Springs
719-262-3500 (fax 3589), wickert@signal.uccs.edu

***Abstract*—The implementation of a low cost, frequency selective, RF channel simulator architecture is explored. The system is implemented almost entirely in the digital domain by use of IF sampling with the signal processing performed in a Texas Instruments TMS320C6701 floating point DSP and a Xilinx XC4044XL FPGA. The prototype system costs less than $3900 and is capable of simulating three delay taps. It is believed that a system capable of simulating 12 delay taps with a bandwidth of 5 MHz could be built at a cost less than $2000, at least an order of magnitude cheaper than commercially available channel simulators of comparable performance.**

I. Summary

Designing a modern wireless communication system is a formidable task. Today's users demand good voice quality, have a low tolerance for busy signals or dropped connections, and desire error free, high-speed data transmission. For a system to efficiently meet these requirements, it must perform well in many different environments where the radio propagation characteristics vary considerably. Ensuring a product's performance requires not only analysis and simulation, but also prototyping and testing. Unfortunately, field-testing a wireless product in all of the possible radio environments is cost prohibitive and time consuming. A much more practical approach is to use a real time channel simulator that may be configured to emulate the various radio propagation characteristics encountered in the real world.

The available channel simulators on the market today are complex and costly (from $24,000 to $500,000 [1]), often requiring several circuit cards and multiple processors. Recent increases in the performance of digital signal processors (DSPs) and field programmable gate arrays (FPGAs) suggest the possibility of greatly reducing the cost and complexity required in implementing a channel simulator. This paper presents the implementation of a low cost real-time frequency selective radio frequency (RF) channel simulator.

II. Background

A mobile radio channel simulation which incorporates both Doppler spreading and delay spread (doubly-spread channel), is typically implemented using a bank of time shifted independent Rayleigh flat fading channel simulators. Two frequently cited techniques for generating Rayleigh flat fading at a prescribed Doppler frequency are, a discrete approximation using sums of sinusoids (*Jakes method* [2]) and quadrature amplitude modulation by independent lowpass filtered (shaped) white Gaussian noise sources [2], [3].

The discrete approximation can produce an extremely accurate estimate of the theoretical Doppler autocorrelation function using sums of sine and cosine generators. This exacting performance comes at the expense of increased computational burden for each flat fading building block. The quadrature amplitude modulation approach obtains its accuracy by choosing a realizable rational transfer function based filter (FIR and/or IIR) to approximate the Doppler power spectrum. Since the power spectrum and autocorrelation function form a Fourier transform pair, the noise shaping filter directly controls the Doppler autocorrelation function. In this work the quadrature amplitude modulation technique is implemented using an efficient noise shaping filter implementation based on [4].

The Doppler frequency referred to above is the maximum Doppler frequency (shift) associated with the true carrier frequency of interest. The maximum Doppler frequency, f_m, is the vehicle velocity times the carrier frequency divided by the speed of light. In the design of the Doppler noise shaping filter described later, f_m in Hz is normalized by the sampling rate.

We assume that the receive antenna is a vertical monopole and the arriving signal rays are uniformly distributed in arrival angle for each Rayleigh bundle. This produces the so-called *bathtub* Doppler power spectrum for the quadrature signals

$$S_{I/Q}(f) = \begin{cases} \frac{3b}{\pi f_m}\left[1-\left(\frac{f}{f_m}\right)^2\right]^{-1/2}, & |f| < f_m \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where b is the mean carrier power level. The corresponding Doppler autocorrelation function is

$$R_{I/Q}(\tau) = \frac{3}{2} b J_0(2\pi f_m \tau) \quad (2)$$

where J_0 is the Bessel function of the first-kind order zero. The digital noise shaping filter, described later in this paper, is designed to approximate the above bathtub spectrum and have a cutoff frequency that corresponds to the desired Doppler frequency (vehicle velocity) that is to be simulated. A good simulator design should be capable of representing vehicle velocities (Doppler frequencies) ranging from a slow walk to freeway speeds.

III. Implementation Issues

The goal of this study was to design, build, and test a proof of concept prototype of a low cost real-time frequency selective RF channel simulator which could accommodate the wide bandwidth of present and future mobile radio systems, provide sufficient time resolution in the specification of delay taps, and allow system configuration flexibility via a Graphical User Interface (GUI). These system objectives present difficult implementation tradeoffs and limitations imposed by several critical design parameters. Those with the biggest impact on the architecture are: the large bandwidth required, the hardware cost, the mobile speed, the interpolation of Doppler spectrum up to intermediate frequency, and the processing power required to implement a sufficient number of delay taps.

The target bandwidth of the system was 5 MHz. This number was deemed appropriate for encompassing the development needs of evolving third generation wireless standards. Building an analog system to accomplish the multitap delay line and fading generators proves to be prohibitively expensive at higher bandwidths. In addition, the accuracy and repeatability increases as more of the design is moved into the digital domain. Thus, an IF sampling architecture was chosen for this design. Implementing a digital system with 5 MHz of bandwidth requires a sampling rate of at least 10 MHz according to the Nyquist sampling theorem [5]. This sampling rate is easily achievable with today's high speed and low cost analog to digital (A/D) converters and digital to analog (D/A) converters. The bottleneck for this system arises from the amount of processing required on the digitized signal. Even with the advances made recently in digital signal processors (DSPs), it is still not feasible to implement the entire frequency selective fading system in a single processor.

A significant portion of the processing power, measured in millions of instructions per second (MIPS), required by the simulator is attributed to the interpolation of the low bandwidth Doppler spectrum up to the intermediate frequency (IF) of the system. Since the Doppler spectrum must be convolved with the input signal spectrum, or multiplied in the time domain, this requires that the two signals be represented in the digital domain at the same rate. In other words, one sample from the fading generator must be multiplied by a corresponding sample of the input signal. The Doppler spectrum for mobile speeds of interest is typically less than 2 kHz wide [2], while the minimum digital IF for a signal bandwidth of 5 MHz is > 2.5 MHz. The resulting interpolation factor is on the order of 1000.

The process of interpolation is described simply as increasing the sample rate of a previously digitized signal by estimating the analog signal's trajectory between the available samples and placing additional samples along this estimated trajectory. This process may be accomplished efficiently and accurately by simply placing zeros between the known samples and filtering the resulting sequence with a properly designed FIR filter as represented by Fig. 1.

One interesting point to note from this process is that the FIR filter must have a memory, or length, long enough

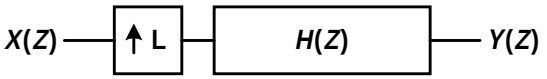

Fig. 1. Traditional interpolation.

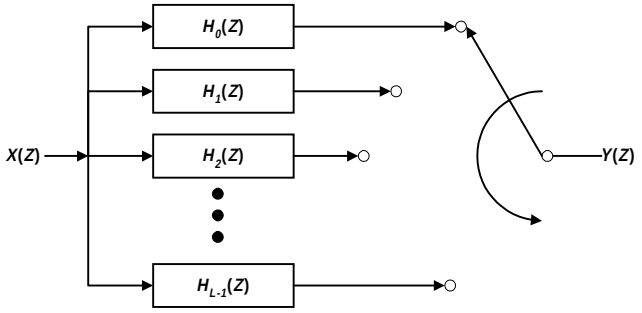

Fig. 2. Polyphase interpolation.

to span the distance between the known samples. In the case of the channel simulator, this would require an FIR filter of minimum length 1000. Using a polyphase filter representation as described in [6] reduces the complexity of this filter considerably. The polyphase filter model is shown in Fig. 2, where the coefficients of the polyphase filters, $h_p[n]$, are related to the coefficients of the filter in Fig. 1, $h[n]$, by

$$h_p[n] = h[nL + p] \tag{3}$$

The advantage of the polyphase structure is that it performs the filtering at the lower sampling rate.

Even with the efficiency improvement of the polyphase filter, implementing an interpolation on the order of L=1000 in real-time is still impractical. A more efficient approach is to perform the interpolation in multiple steps [7]. This reduces the complexity to a more reasonable level for realization in a DSP chip. Thus, designing the interpolation becomes an exercise in optimizing the number and lengths of the interpolation stages. Specifically, for the channel simulator, the number of processor cycles required to perform the interpolation is dictated by the inner loop of the polyphase FIR interpolation filter. This loop is described by

$$y[n] = x[k]h[p] + x[k-1]h[L+p] \tag{4}$$

where n is the sample index after interpolation and k is the sample index before interpolation. The execution of this loop in a DSP requires 4 loads, 2 additions, 2 multiplies, and 1 store; for a total of 9 instructions. Assuming that today's very long instruction word (VLIW) DSP architectures can execute some of the instructions in parallel (a complicated issue determined by factors such as pipeline delays and allocation of instructions to separate arithmetic logic units), this operation could be executed in as few as 2 cycles. This absolute best case is for the fastest available processor, the Texas Instruments TMS320C6701, which has 8 parallel execution paths, two of which may perform multiplies [8]. This assumes that the pipeline is kept

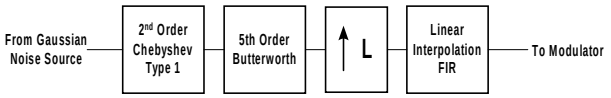

Fig. 3. Shaping filter realization.

full 100% of the time. A more realistic, yet still somewhat optimistic, estimate would place the number of cycles at 4. Given that each tap delay has both an in phase (I) and quadrature (Q) component, the estimated number of cycles per tap for the inner loop of the FIR is 8.

The number of CPU cycles per output of the fading generator, if implemented in DSP, approaches the number of cycles for the inner loop, 8, in the best case. This is shown as

$$c = \frac{r}{\prod_{i=0}^{n-1} L_i} + \frac{m}{\prod_{i=1}^{n-1} L_i} + \frac{m}{\prod_{i=2}^{n-1} L_i} + \cdots + m \qquad (5)$$

where r is the number of cycles for the random number generator, n is the number of interpolation stages, m is the number of cycles in the inner loop, and L is the interpolation factor for each stage. For example, if two interpolation stages are used, the number of cycles per output is

$$c = \frac{r}{L_0 L_1} + \frac{m}{L_1} + m \qquad (6)$$

Thus, in the best case, assuming a reasonable number for r, c approaches m. From the above, m is 8 in the best case for this system. Since the system clock is 160 MHz and the minimum sampling rate for the desired bandwidth is 10 MHz, this means that each fading generator must output a value every 16 clock cycles. This analysis constrains the number of taps in the multipath model to 2 in the very best case if the DSP must do all of the processing. This result is what led to the inclusion of a field programmable gate array (FPGA) in the system to increase the number of available MIPS. The FPGA accelerates the interpolation and multiplication processes by implementing them in parallel.

IV. The Prototype Architecture

This protoype is based on a paper by Wickert and Jaycobsmeyer which describes a software simulation of the QAM method using IIR Doppler shaping filters and FIR linear interpolation filters as shown in Fig. 3 [4]. The cascaded Chebyshev and Butterworth shaping filter was chosen based on the requirements of the TIA IS-55A specification [9]. The output of the shaping filter is upsampled and interpolated to reduce the number of calculations needed in the simulation. This variation of the QAM method was chosen for implementation due to its computational efficiency.

A thesis by C. M. Keen implements a flat fading software simulation of the Wickert/Jaycobsmeyer method in the C language on a PC platform [10]. The Chebyshev/Butterworth filter synthesis is described in detail in this reference. To summarize, the digital IIR filter is derived from an analog prototype using the bilinear transform. The spectral moments, b_o and b_2 of the resulting filter, are calculated using

$$b_n = (2\pi)^n \int_0^{1/2} |H(e^{j2\pi f})|^2 f^n \, df \qquad (7)$$

where $H(e^{j2\pi f})$ is the transfer function of the filter with f being the normalized digital frequency in cycles/sample. These moment values are used to calculate the sampling rate normalized (maximum) Doppler frequency, f_m, according to

$$f_m = \frac{1}{2\pi} \sqrt{\frac{2b_2}{b_0}} \qquad (8)$$

for a particular digital filter cutoff frequency, f_c. If f_m from (8) is not within the desired tolerance, then the cutoff frequency of the digital filter is changed and a second iteration commences. Iteration, via a bisection algorithm, continues until a filter giving an acceptable f_m results. The filter synthesis portion of Keens program is adopted and modified to work under the Microsoft Windows environment for this prototype.

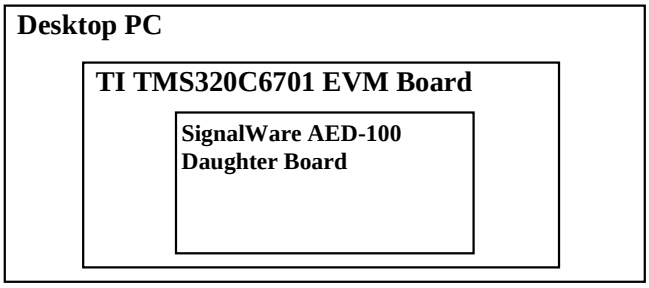

Fig. 4. System components.

The system architecture shown in Fig. 4 consists of three hardware components. A standard personal computer (PC) provides the Doppler shaping filter synthesis, amplitude scaling, and the graphical user interface (GUI). A Texas Instruments TMS320C6701 EVM DSP board which connects to the PC's PCI bus is responsible for the random number generator, Doppler spectrum shaping filter, and the first interpolation stage. An AED100 daughterboard from Signalware contains the A/D and D/A components as well as the FPGA for the tapped delay line, multipliers, and final interpolation stage of each fading generator. A detailed block diagram of the system appears in Fig. 5.

The system employs bandpass sampling of a 70 MHz intermediate frequency signal with up to 5 MHz of bandwidth. Bandpass sampling theory states that a signal may be sampled with a rate that is much less than the highest frequency component of the input signal without aliasing, provided that [11]

$$nf_s + B \leq 2f_o \leq (n+1)f_s - B \qquad (9)$$

where n is a positive integer, f_o is the center frequency of the input signal, f_s is the sampling frequency, and B is

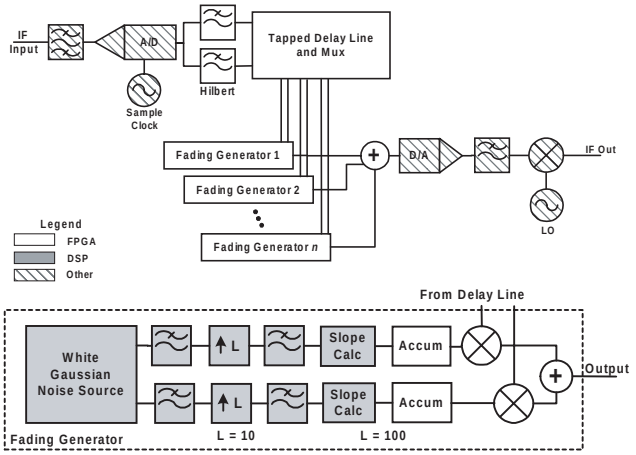

Fig. 5. System architecture.

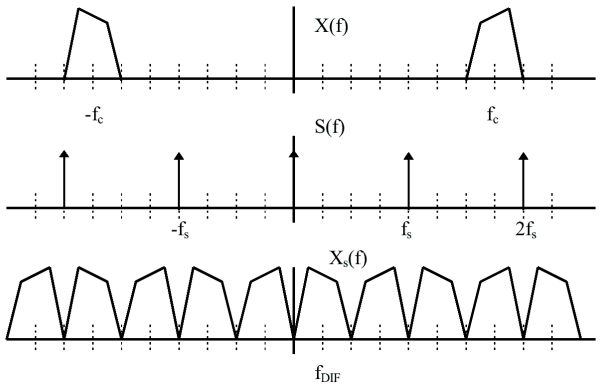

Fig. 6. Bandpass Sampling (Reproduced with permission from [11]).

the bandwidth of the signal. This under sampling process effectively translates the input signal to a digital IF which is an alias of the input signal as shown in Fig. 6.

For this design, the sample clock is 11.1 MHz, thereby setting the input frequency range from 66.6 MHz to 71.6 MHz, with a digital IF of 2.5 MHz. An analytic signal is derived by splitting the signal into two paths and Hilbert transforming one path with an asymmetric FIR filter while introducing the equivalent group delay in the second path via an FIR all pass filter.

The resulting analytic signal enters the tapped delay line where user selected delays are introduced. The delayed signals are each multiplied in the time domain by the independent fading generator signals as shown in Fig. 5. The fading generator consists of the random number generator, two interpolation stages, and the modulator. The random number generator creates two white, uncorrelated, Gaussian random number sequences. This method is based on the C compiler's built in `rand()` function as described in [12]. The `rand()` function generates uniformly distributed random integers, which are converted into a Rayleigh distributed random variable by inverting the Rayleigh cumulative distribution function to obtain

$$R = \sqrt{-2\sigma^2 \log_e(1 - \texttt{rand()}/\texttt{RMAX})} \tag{10}$$

where σ^2 sets the variance of the Rayleigh variates and `RMAX` is the maximum random number generated by the `rand()` function. The resulting uncorrelated outputs are generated using

$$X = R\cos\theta \quad \text{and} \quad Y = R\sin\theta \tag{11}$$

where θ, a uniform on $[0, 2\pi]$ random variate, is also generated using the `rand()` function. To speed up the implementation, a table of R values is stored in the EVM memory along with cosine and sine tables.

Each output from the random number generator has its spectrum shaped by an IIR filter to approximate the effects of Doppler spreading. The classical Doppler spectrum derived in [2] is approximated according to the guidelines defined in [9]. The mobile speed and carrier frequency parameters are input by the user through a Windows-based GUI. The filter coefficients for the Doppler shaping IIR filters are calculated on the PC and downloaded over the PCI bus to the DSP card.

The first interpolation stage is accomplished in the DSP by using the FIR polyphase representation of a linear interpolation filter. The impulse response of the interpolation filter is the triangle sequence given by $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1\}$. The response of this filter is designed to have nulls corresponding to the images resulting from the introduction of zeros in the interpolation process.

The second interpolation stage is also a linear interpolator. Because of the high sampling rate, this stage is implemented partially in the FPGA. The impulse response for the second interpolation stage is of the same form as that in the first stage, but is realized differently. The DSP calculates a slope value between the previous and current samples output from the first interpolation stage and writes this slope value to the FPGA. The FPGA increments an accumulator by the slope value on each sample clock cycle to obtain the output signal (See Fig. 5). Fig. 7 shows the results of a simulation of the interpolator structure. This spectral plot shows the upsampling leakage that results from the non-ideal interpolation filter. The leakage components are down about 60 dB.

Once the interpolation of the Doppler spectrum is complete, the resulting in phase and quadrature components are multiplied by the corresponding components of the sampled and delayed input signal. The I and Q products sum to complete one tap of the multipath simulator. Each additional tap repeats this process with the outputs being summed together immediately before the D/A converter. A third-order Bessel low pass filter is used for reconstruction. It is worth noting that the flexibility of the architecture allows reconfiguring the system for use with Doppler spectra other than the classical simply by reprogramming the filter coefficients. Furthermore, the system hardware is not limited to using the QAM method. It is feasible that the popular Jakes method could be used as well.

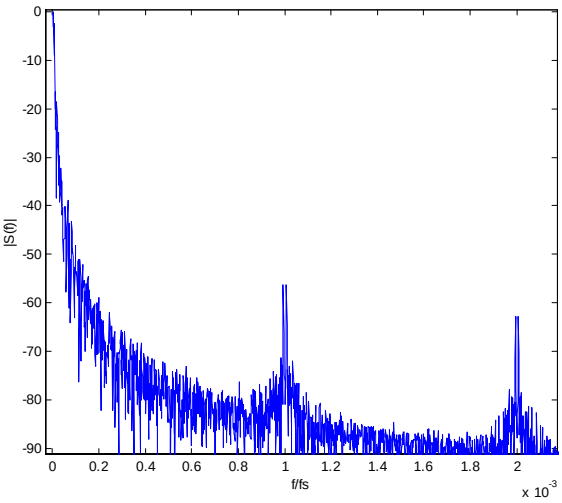

Fig. 7. Interpolation Simulation First Stage in DSP , Second Stage in Matlab.

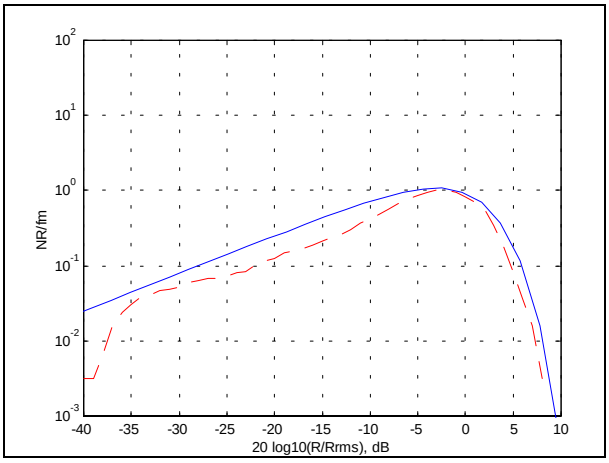

Fig. 8. Measured (dashed) vs. theoretical (solid) level crossing rates.

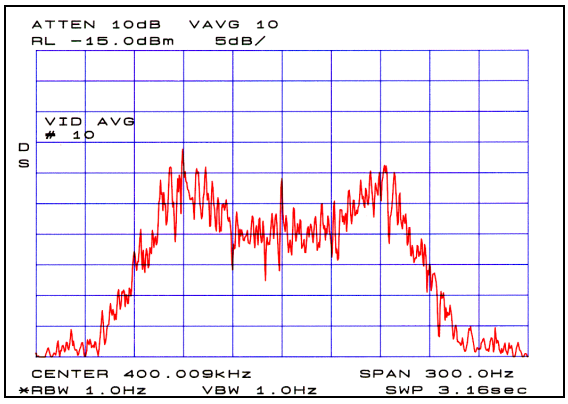

Fig. 9. Measured Doppler spectrum.

V. System Performance

The prototype system has a bandwidth of approximately 5 MHz with a delay tap spacing of less than 100 nanoseconds. The fading statistics of each tap compare favorably with test specifications in the TIA IS-55A performance standard for mobile stations [9]. The measured level crossing rate is compared to the theoretical in Fig. 8. The dynamic range of the prototype is limited by the 8-bit A/D and D/A converters. More dynamic range is desirable because the multipath environment typically causes fades greater the 40 dB. This could easily be implemented by integrating 10 or 12 bit A/D and D/A converters. The prototype is limited to three delay taps due to the amount of logic in the FPGA. This prototype uses a Xilinx XC4044XL with a maximum of 80,000 logic gates. This component is at 97% capacity with three taps and maximum delays of about 20μs. Given this limitation of FPGA resources, the delay spacing is not dynamically reconfigurable on the prototype. A new FPGA file must be loaded into the FPGA for each desired delay configuration. These limitations could be overcome by upgrading the FPGA on the daughterboard. The state of the art Xilinx Virtex FPGA has a capacity greater than 4 million gates. With this factor of 50 increase in capacity, it is safe to assume that 12 configurable tap delays could be implemented in the FPGA. The DSP has plenty of available bandwidth to accommodate 12 taps.

Fig. 9 shows the measured Doppler spectrum for a simulated signal with mobile velocity of 100 kph and carrier frequency of 850 MHz. The 3 dB bandwidth is approximately 150 Hz and the peak is approximately 60 Hz from f_o, which is roughly 75% of f_m. These values are consistent with the desired results.

The characterization of the delay properties of the channel simulator is based on the sliding correlator measurement system [5]. Since a spread spectrum sliding correlator measurement system with the desired frequency range was not available for use in this testing, a system using a GSM transmitter and receiver was implemented. The transmitter outputs GSM access bursts during one timeslot of each frame. Each access burst contains a training sequence with good correlation properties which is used for timing advance measurements in the GSM reverse link [13]. A hardwired signal from the transmitter to the receiver provides timing synchronization of the bursts. The receiver samples the GSM signal at the baud rate and correlates the received samples with the known training sequence. As in the spread spectrum sliding correlator, the resulting correlation shows the delay profile of the channel.

Fig. 10 shows a waterfall plot of 10 consecutive bursts, with the channel simulator configured for three taps with power levels of 0, -6, and -12 dB. The second tap is configured for an excess delay approximately 14.5μs. The third tap is configured for an excess delay of approximately 29μs.

VI. Conclusions

The cost of the prototype components appears in Table 1. The cost of the RF sections is included for comparison with other available channel simulators. This RF

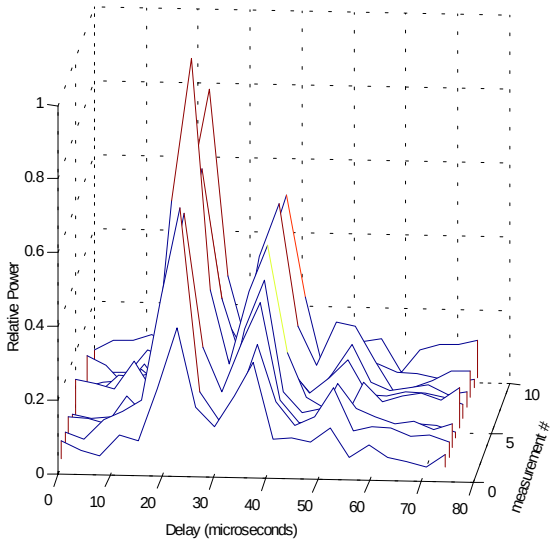

Fig. 10. Delay spread characterization.

TABLE I

PROTOTYPE HARDWARE COST.

Component	**Cost**
Texas Instruments TMS320C6701 EVM	$1495
Signalware AED-100 Daughter board	$995
XC4044XL FPGA Option for AED-100	$810
SAW Filter	$100 (Est.)
RF Sections	$500 (Est.)
Total Prototype Hardware Cost	**$3900**

section cost was conservatively estimated based on discussions with the expert radio designers at Xircom Wireless Technology Group (formerly Omnipoint Technologies).

This prototype implementation has some inefficiency due to the number of unused parts included in both the DSP board and the daughterboard. Integrating the system onto one custom printed circuit board and eliminating unused parts could reduce the cost considerably. It is believed that this system could be built in small quantities for as little as $2000 per unit. This does not include the cost of the PC or the signal generator used for the local oscillator, which are common items in most engineering laboratories. When compared to the cost of the commercially available channel simulators, this is a reduction of at least one order of magnitude.

REFERENCES

[1] B. Schweber, "RF-Channel Simulators: Bring Realitys Challenges to Your Prototype," pp. 52–63, EDN, September 11, 1998.

[2] W. C. Jakes, *Microwave Mobile Communications*, IEEE Press, 1974.

[3] T.S. Rappaport, *Wireless Communications Principles and Practice*, Prentice Hall PTR, 1996.

[4] M. A. Wickert and J. M. Jaycobsmeyer, "Efficient Rayleigh Mobile Channel Simulation Using IIR Digital Filters with Upsampling," *Proceedings ICSPAT*, pp. 391-396, Oct. 1995.

[5] Alan V. Oppenheim and Ronald W. Schafer with John R. Buck, *Discrete-Time Signal Processing*, second edition, Prentice Hall, Upper Saddle River, NJ, 1999.

[6] f. harris, "Multirate Digital Signal Processing in Digital Receivers," Keynote Presentation at *Second International Symposium on DSP for Communication Systems* (DSPCS-94), Adelaide, Australia, April, 1994.

[7] R. E. Crochiere and L. R. Rabiner, "Interpolation and Decimation of Digital Signals - A Tutorial Review," *Proceedings of the IEEE*, vol. 69, no. 3, Mar. 1981.

[8] Texas Instruments, *TMS320C62x/C67x CPU and Instruction Set Reference Guide*, Texas Instruments Literature Number: SPRU189C, March, 1998.

[9] TIA IS-55A, "Recommended Minimum Performance Standard of 800 MHz Dual-Mode Mobile Stations," Sept, 1993.

[10] C. M. Keen, "C Language Program Simulator for a Rayleigh Fading Channel," A thesis submitted to the Graduate School of the University of Colorado at Colorado Springs, Fall 1994.

[11] B. L. Fox, "Analysis and Dynamic Range Enhancement of the Analog-to-Digital Interface in Multimode Radio Receivers," Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University, Blacksburg, VA, Feb. 1997.

[12] S. A. Tretter, *Communication System Design Using DSP Algorithms with Laboratory Experiments for the TMS320C30*, New York and London: Plenum Press, 1995.

[13] S. M. Redl, M. K. Weber, and M. W. Oliphant, *An Introduction to GSM*, Boston: Artech House, 1995.